1J-7

大規模分散ファイルシステム向きの高機能ストレージシステムの研究

山下 直人† 小柳 順裕† 田胡和哉†

東京工科大学†

１． はじめに

大規模な分散ファイルシステムのキャッシュに用いることができる外部記憶装置の構成方法について提案する。

外部記憶装置は、OSD(Object-based Storage Devices)規格によって接続する。OSD 規格によれば、ブロック単位ではなく、ブロックの集合(Object)単位でのアクセスが可能になる。Object ごとのアクセス特性を把握することにより、自己最適化機能を導入しやすい特徴を持つ。

ここでは、キャッシュ記憶装置内部で Object ごとのアクセス特性を統計的に解析し、性能の自己最適化を行うキャッシュ記憶装置を構築することを試みる。本稿では、その構造、アルゴリズムについて述べる。

２． システム構成

２．１ リソース

高機能ストレージシステムは、図１のような構成になっている。キャッシュの各々にストレージシステムが接続され、ストレージシステムは１ドライブ、メモリ、組み込み用プロセッサで構成され、玄人志向の NAS 組み立てキット、玄箱のようなものを考えている。

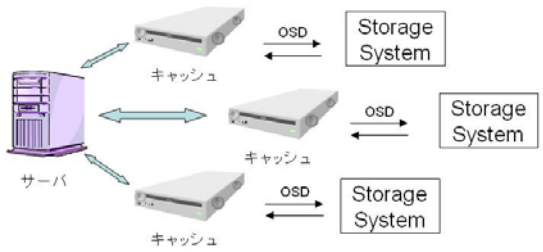

図１ システムの構成図

２．２ OSD

ストレージアクセスのためのプロトコルとして Object-Based Storage Device (OSD)を用いる。OSD とは、ファイルシステムの一部機能を外部にオフロードするための規格で、ファイルのメタデータのやりとりの際に OSD を用いることによって、少ないオーバヘッドでスケーラブルな性能を持つキャッシュノード向きストレージデバイスを実現することができる。

３．スケジューラ実装に対するアプローチ

実現するストレージ装置の性能の最適化を図る機構を、スケジューラとよぶことにする。スケジューラの実装にあたって、高い性能を得るとともに、自動的な性能改善が図られる構造をとることが目標である。

３．１ ポリシー階層化

システムが自らの機能によって性能改善を図る機構の実現を試みるアプローチとして、スケジューラを大きな単一のプログラムとして実現するのではなく、単純なスケジューリング目標を達成する複数の小規模スケジューラの集合からなるモジュール構造とすることを試みる。それらを要素スケジューラとよぶことにする。要素スケジューラは個々にスケジューリング目標を持つ。スケジューラ全体の自動最適化を図るよりも、要素スケジューラの自動最適化の方が問題が単純化される。特に、主記憶等の単一のシステム資源を、複数の要素スケジューラの間で競争的に共有し、それらへの資源配分を決める上位の要素スケジューラを実現する構造をとることにより、複雑なスケジューラを構造的に構築することが可能になる。これを、ポリシー階層化とよぶことにする。

図２に、２階層からなるスケジューラの構造の例を示す。Child Scheduler が下位の要素スケジューラに対応し、Parent Scheduler が上位の要素スケジューラに対応する。Child Scheduler は、それぞれにスケジューリング目標を持つ。この例では、一方の Child Scheduler が単位スループット当りの I/O 回数の削減を目標としており、主記憶によるディスクブロックのキャッシング機構に対応する。もう一方は I/O 効率(ディスク I/O の実効スループット)の向上を目標としており、先読み機構に対応する。両者は、主記憶の獲得において競合関係にある。どちらにどれだけ主記憶を割り当てると最適な全体性能が得られるかは、Parent Scheduler が決める。

Proposal of a High Efficiency Storage System for Large-scale Distributed File Systems

†Naoto Yamashita, Masahiro Koyanagi, Kazuya Tago

・Tokyo University of Technology

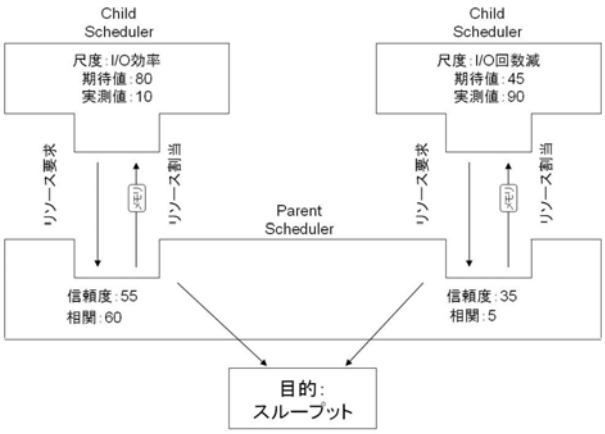

図２　２階層スケジューラ

この構造は、プログラムの構造に対応している。Child Scheduler は、同一のフレームワークにそって、それぞれ独立したプログラムとして実現される。個々の Child Scheduler は、目標を表す識別子(尺度)、その目標値の現在値(実測値)、資源の追加割り当てによって実測値が改善される予想値(期待値)を定義する。Parent Scheduler は、個々の Child Scheduler の目標と、自らの目標の間の相関を測定し、これに期待値を加重することによって各 Child Scheduler への資源割り当てを決める。

３．２　ディスク物理入出力効率の最適化

ディスクの物理入出力効率の最適化を図るには、spatial locality の向上が必要である。すなわち、同時にアクセスされることの多いファイルは、ディスク装置の記憶媒体上でも近い位置に配置されている必要がある。このためには、ファイルイメージをディスク装置上で移動したりコピーしたりする方法が試みられてきた。しかしながら、同時にアクセスされるファイル間の関係が、偶然に起因するものと、必然的なものがあり、それらを弁別せずにこのような再配置を行うと効果が制約される問題がある。ここでは、

１）　必然的に同時アクセスされる可能性の高いファイル群を統計的に抽出してディスク装置上のゾーンを複数構成する

２）　物理入出力のためのバッファとして仮想記憶を利用し頻繁なファイル移動を効率よく実現するとともに、そのページスワップ領域をゾーン内に置くことによって偶然に起因する access locality においても spatial locality が自動的に改善されるようにする

方式をとる。

アプリケーションの種類ごとに、ディスク装置上にゾーンが形成されることが期待できる。たとえば、テキスト編集では、アプリケーション、ライブラリ、文書データが、短期間でアクセスされることが期待できる。同様に、プログラム作成、ブラウザの動作でも、短期間でアクセスされるファイル群が存在する。これらのアクセスの特徴は、必然的な理由がある。一方、２台のクライアントノードで、同時にテキスト編集とシステムビルドが別々に実行されている際、テキスト文書とコンパイラの参照は偶然によるものである。そこで、テキスト文書に関連したゾーン中にバッファを設け、テキスト文書に関係しないファイルを集めることによって、単位時間内に実行される積算シーク距離を短くすることができる。このバッファは、仮想記憶のスワップ領域として実現することによって少ないオーバヘッドで、ダイナミックに変化するアクセスパターンに追従したディスク上のページ移動を実現できる。

４．実装状況

Linux 上で稼動する、OSD サーバ(記憶媒体に RAW Device)のフレームワークが完成し、クライアントとして動作するベンチマークツールを実装している。Linux カーネルモジュールとして実装された OSD ファイルシステムの基本実装が完了している。

スケジューラのフレームワークの基本実装(Parent Scheduler、Child Scheduler、資源のベースクラス)が完了し、それらを利用した、単純な資源配分を行う Parent Scheduler、RAM スケジューラ、が完成している。プリフェッチスケジューラは実装中でこれらの Child Scheduler は、OSD サーバの中に組み込み実装している。今後、評価結果について報告したい。

参考文献

[1]小柳順裕,山下直人,田胡和哉:キャッシュサーバを用いた大規模分散ファイルシステムの構築と応用,第 68 回全国大会

[2]Ralph O. Weber:Object-Based Storage Device Commands (OSD),http://www.t10.org~/ftp/t10/drafts/osd/osd-r10.pdf

[3]S.Jiang, X.Ding, F.Chen, E.Tan, X.Zhang: DULO An Effective Buffer Cache Management Scheme to Exploit Both Temporal and Spatial Locality,File and Storage Technologies (FAST'05)

[4]H.Huang, W.Hung, Kang G. Shin: FS2 Dynamic Data Peplication in Free Disk Space for Improving Disk Performance and Energy Consumption, SOSP(2005)